



Projet 7 : Implémentez un modèle de scoring

Oumeima EL GHARBI

OpenClassrooms – Data Scientist

Projet démarré le : 28/11/2022 sur le projet archivé

Soutenance : 22/02/2023

Plan

Introduction

- Problématique
- Analyse exploratoire

I. Modélisation

- Métriques et fonction coût métier
- Méthode d'entraînement
- Résultats et choix du modèle

II. Dashboard

- Back-end : FastAPI
- Front-end : Streamlit

III. Déploiement

- Heroku app
- Démo

Conclusion

Introduction



Problématique :

«Vous êtes Data Scientist au sein d'une société financière, nommée « Prêt à dépenser » qui propose des crédits à la consommation pour des personnes ayant peu ou pas du tout d'historique de prêt.

L'entreprise souhaite mettre en œuvre un outil de “**scoring crédit**” pour **calculer la probabilité qu'un client rembourse son crédit, puis classifie la demande en crédit accordé ou refusé**. Elle souhaite donc développer un algorithme de classification [...]

De plus, les chargés de relation client ont fait remonter le fait que les clients sont de plus en plus demandeurs de **transparence** vis-à-vis des décisions d'octroi de crédit. [...] Prêt à dépenser décide donc de développer un **dashboard interactif** pour que les chargés de relation client puissent à la fois expliquer de façon la plus transparente possible les décisions d'octroi de crédit, mais également permettre à leurs clients de disposer de leurs informations personnelles et de les explorer facilement.

Implémentation :

Cadre :

Data science : Machine Learning, Classification

Data engineering : API et dashboard

MLOps : Cloud deployment

Modèles testés :

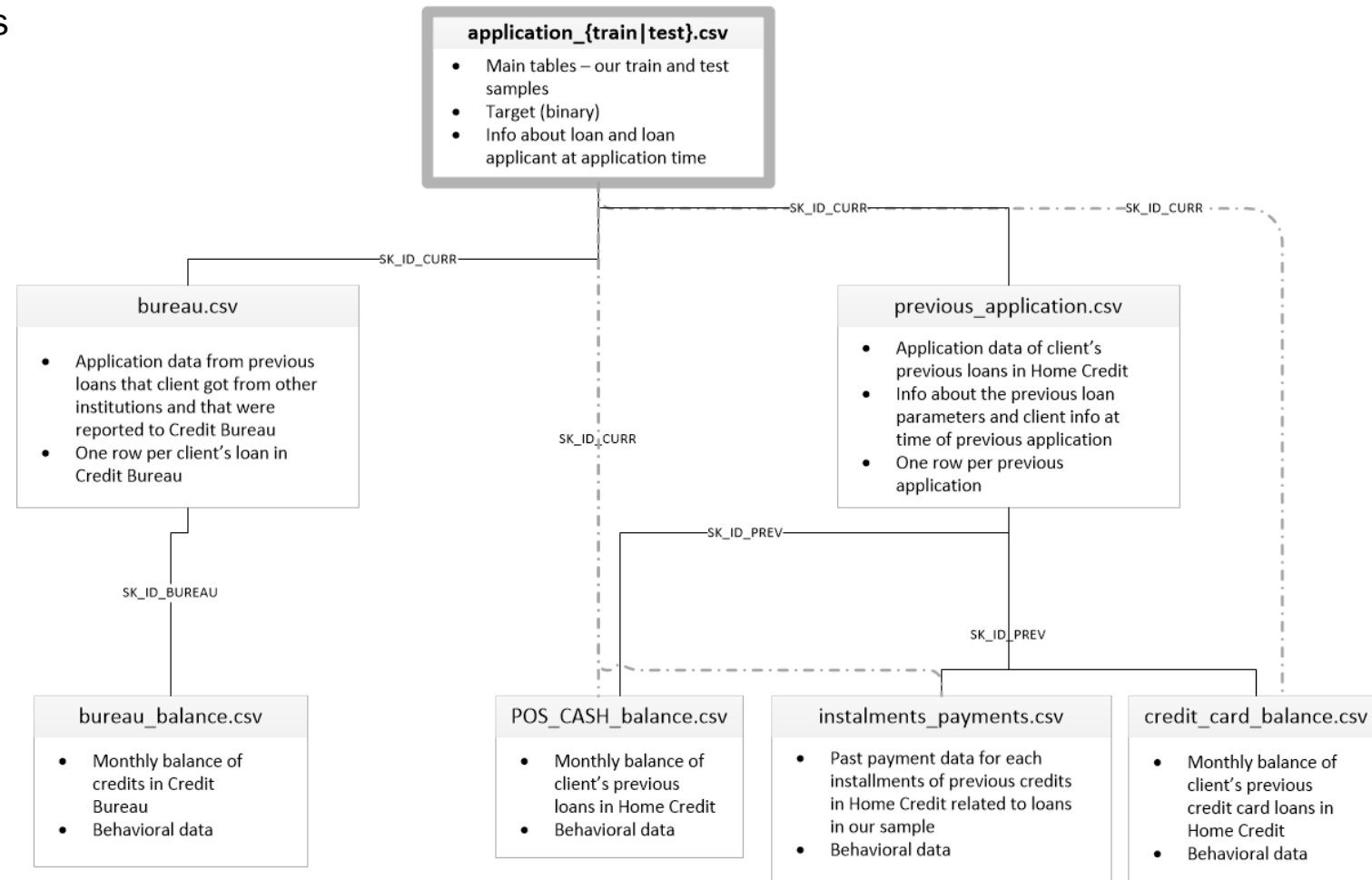
- Dummy
- Regression Logistique
- Random Forest
- HistBoost
- LightGBM

Evaluation :

- AUC-ROC
- F beta score
- Recall
- Precision

Jeu de données

- application_test.csv : notre base de données de nouveaux clients => dashboard
- HomeCredit_columns_description.csv => dashboard
- application_train.csv ; fichier de nouvelles demandes de prêts
- bureau.csv
- bureau_balance.csv
- credit_card_balance.csv
- installments_payments.csv
- POS_CASH_balance.csv
- previous_application.csv



Jeu de données

Jeu de données déséquilibré : 92% de classe 0

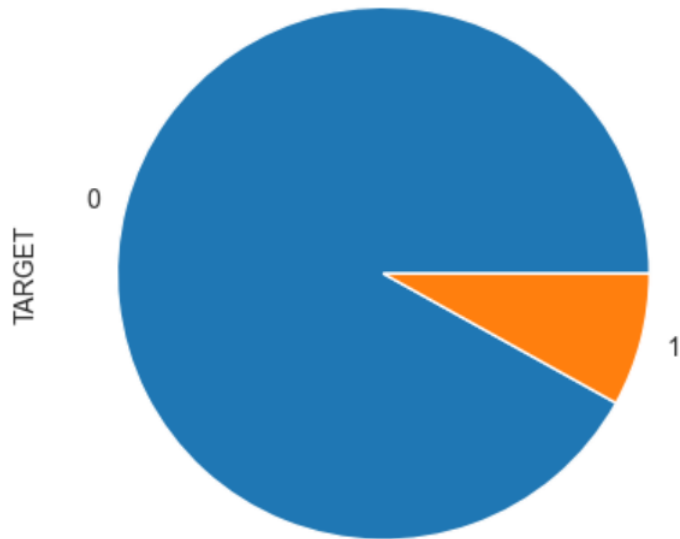
=> Rééquilibrage avec le sur-échantillonnage SMOTE

Shape dataset

The dataset called : dataset_application_test has a shape : (48744, 121)
The dataset called : dataset_application_train has a shape : (307511, 122)
The dataset called : dataset_bureau has a shape : (1716428, 17)
The dataset called : dataset_bureau_balance has a shape : (27299925, 3)
The dataset called : dataset_credit_card_balance has a shape : (3840312, 23)
The dataset called : dataset_installments_payments has a shape : (13605401, 8)
The dataset called : dataset_POS_CASH_balance has a shape : (10001358, 8)
The dataset called : dataset_previous_application has a shape : (1670214, 37)

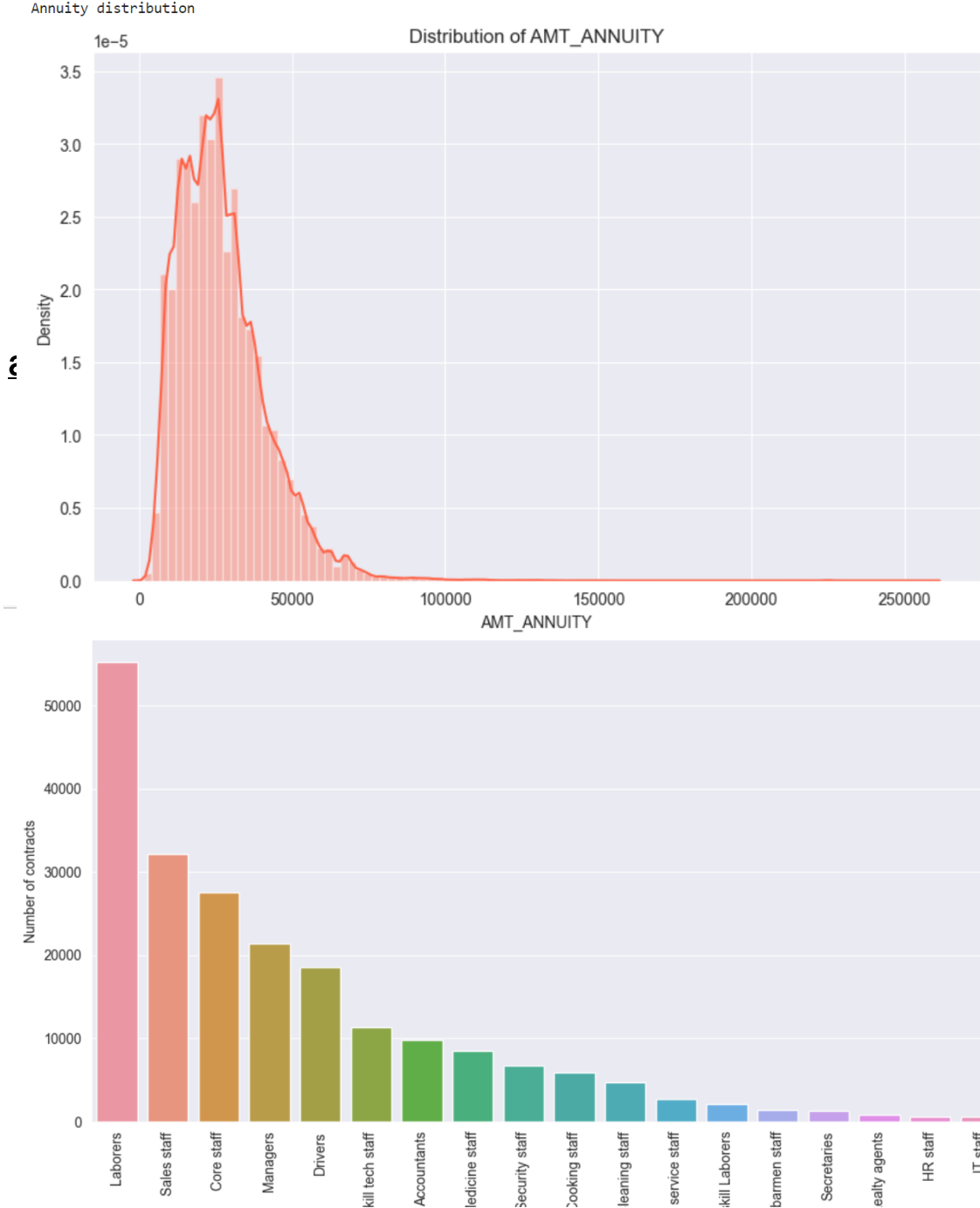
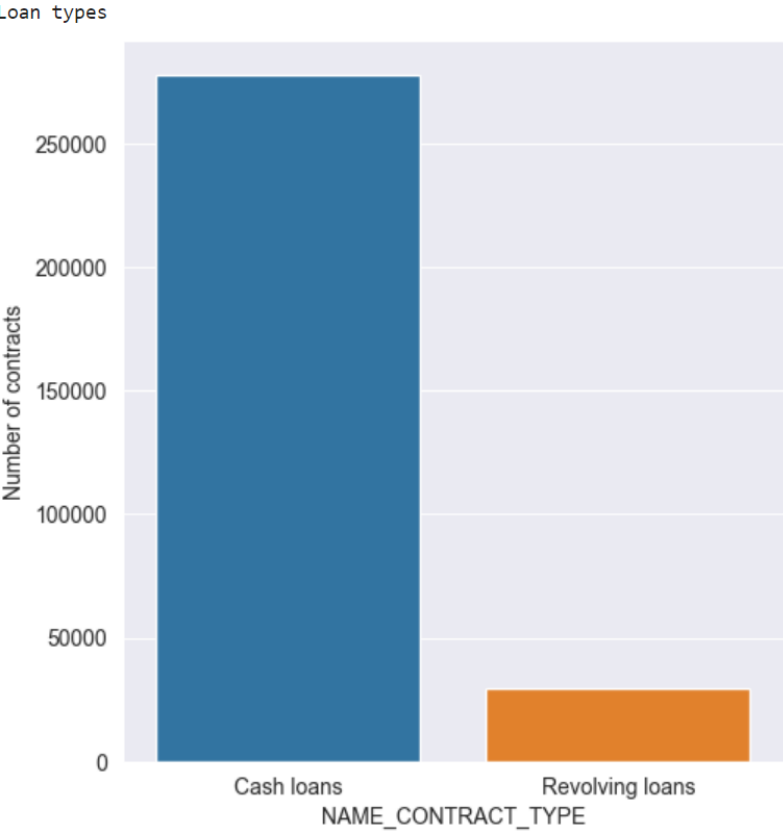
Missing values

The dataset called : dataset_application_test has : 24 % of missing values.
The dataset called : dataset_application_train has : 24 % of missing values.
The dataset called : dataset_bureau has : 14 % of missing values.
The dataset called : dataset_bureau_balance has : 0 % of missing values.
The dataset called : dataset_credit_card_balance has : 7 % of missing values.
The dataset called : dataset_installments_payments has : 0 % of missing values.
The dataset called : dataset_POS_CASH_balance has : 0 % of missing values.
The dataset called : dataset_previous_application has : 18 % of missing values.
The dataset called : dataset_sample_submission has : 0 % of missing values.





Analyse exploratoire

Analyse exploratoire des jeux de données : application_train, bureau, previous_application



I) Modélisation

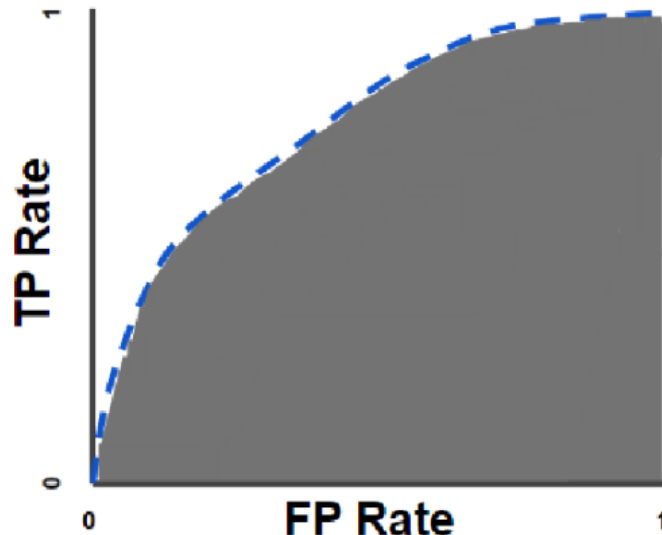
- 
- 
- Métriques et fonction coût métier
 - Méthode d'entraînement
 - Résultats et choix du modèle

Modélisation

1) Métriques et fonction coût métier

- Fonction coût métier : F bêta score où $\beta = 2$ (valeur de β dépend de la logique métier)
- Recall à maximiser : on préfère ne pas prêter à de mauvais clients que de perdre de potentiels bons clients
- Accuracy non retenue (car les classes sont déséquilibrées)
- AUC retenue

AUC: Area Under the ROC Curve



	Predicted	
	Negative	Positive
Actual Negative	True Negative	False Positive
Actual Positive	False Negative	True Positive

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

Modélisation

2) Méthode d'entraînement

Problème de classification binaire

- Prédire 0 => le client rembourse le prêt : le crédit est accordé par la banque
- Prédire 1 => le client risque de ne rembourser pas le prêt : le crédit ne sera pas accordé

- Modèles :

- Dummy : Accuracy à 92% => prédit toujours la classe majorité

- Logistique Régression : Standardisation des valeurs non encodées
- Random Forest
- HistBoost
- LightGBM

Modélisation

2) Méthode d'entraînement Gridsearch et Cross-Validation

- Application_train => train and test set
- Evaluation sur test set (X_test et y_test)
- Train set utilisé en validation croisé pour l'entraînement avec X_train et X_validation
- Stratified K Fold (K = 5)
- Pour le LightGBM (K = 10)

Recherche des meilleurs hyper-paramètres avec GridSearchCV pour Régression Logistique, HistBoost, Random Forest

- ⇒ Problème de performance du fait d'un temps de calcul élevé
- ⇒ LightGBM non fine-tuné

Modélisation

2) Méthode d'entraînement

Oversampling : SMOTE (Synthetic Minority Oversampling Technique)

Sur-échantillage = dupliquer des échantillons de la classe minoritaire

Méthode SMOTE pour rééquilibrer le dataset

Utilisé pour :

- Régression Logistique
- Random Forest
- HistBoost

- Résultats : légère amélioration des score (AUC, Recall) mais amélioration négligeable

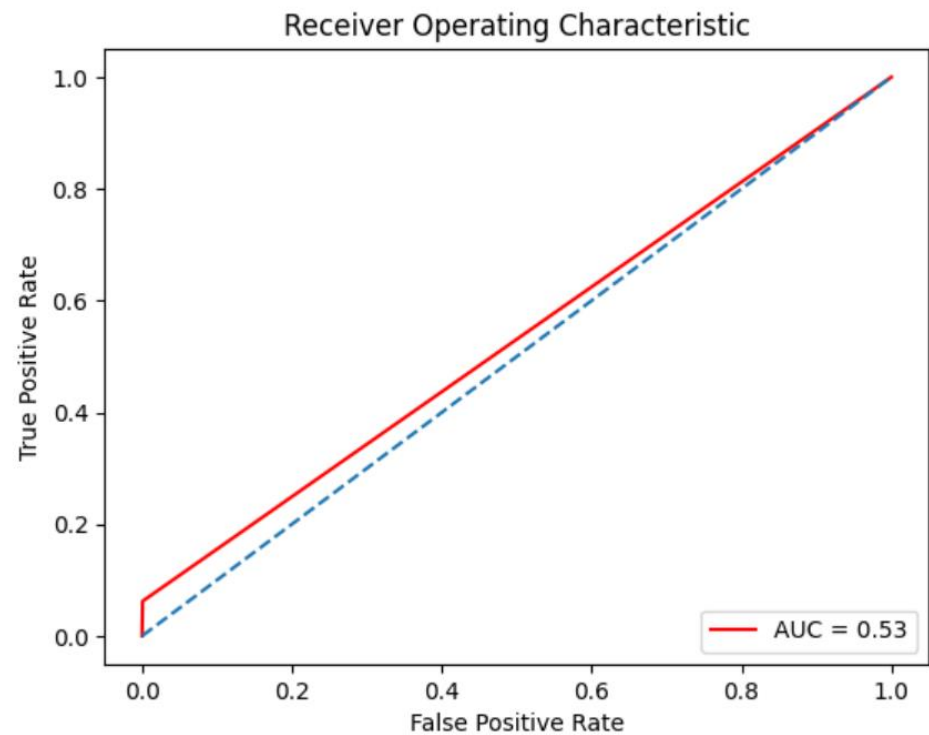
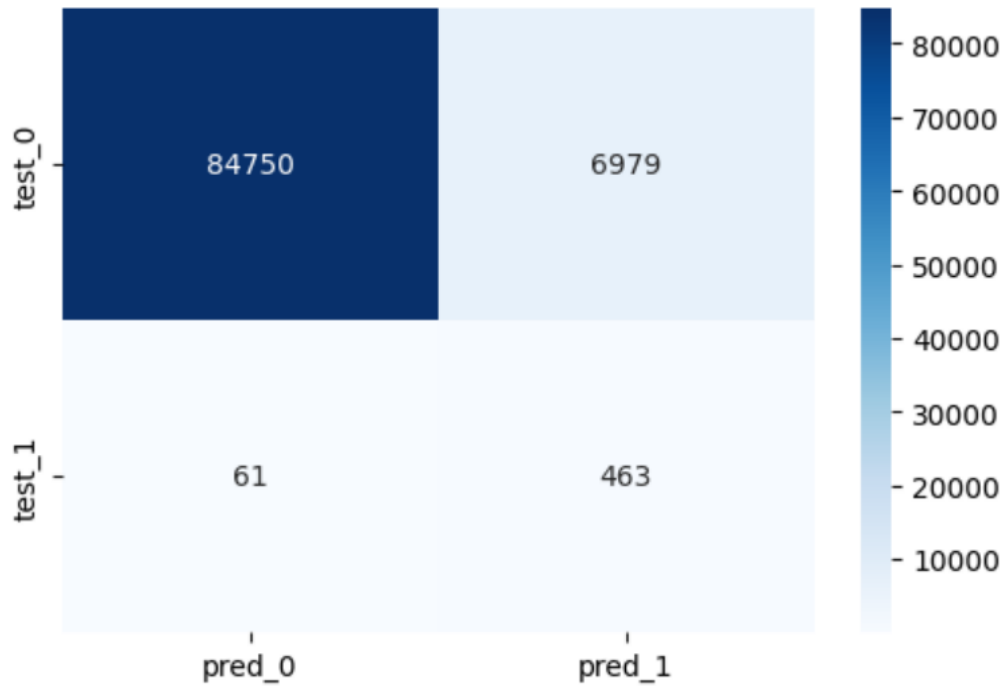
- LightGBM non sur-échantillonné car temps d'entraînement élevé et SMOTE apporte une amélioration négligeable des résultats

__Model__	__ROC-AUC__	__F-score Beta = 2__	__Recall__	Precision	F1-score	F1-score weighted	Accuracy
Dummy	0.500	0.000	0.000	0.000	0.000	0.881	0.919
log_reg_SMOTE_1	0.495	0.301	0.981	0.080	0.148	0.028	0.087
log_reg_SMOTE_2	0.495	0.301	0.981	0.080	0.148	0.028	0.087
log_reg_3	0.494	0.301	0.980	0.080	0.148	0.026	0.086
log_reg_2	0.493	0.300	0.972	0.080	0.147	0.036	0.091
log_reg_SMOTE_3	0.493	0.300	0.976	0.080	0.147	0.031	0.088
log_reg_SMOTE_4	0.492	0.299	0.970	0.079	0.147	0.038	0.091
log_reg_1	0.489	0.297	0.960	0.079	0.146	0.044	0.094
log_reg_SMOTE_0	0.488	0.296	0.943	0.079	0.146	0.071	0.107
log_reg_4	0.481	0.288	0.894	0.078	0.143	0.128	0.135
log_reg_0	0.479	0.285	0.872	0.077	0.142	0.155	0.149

Modélisation

3) Résultats et choix du modèle

Modèle
retenu :
LightGBM



Prediction for : TARGET

	__Model__	__ROC-AUC__	__F-score Beta = 2__	__Recall__	Precision	F1-score	F1-score weighted	Accuracy
0	LightGBM_7	0.531	0.076	0.062	0.884	0.116	0.892	0.924
0	LightGBM_8	0.523	0.057	0.046	0.842	0.087	0.889	0.922
0	LightGBM_4	0.523	0.057	0.047	0.834	0.088	0.889	0.922
0	LightGBM_1	0.521	0.053	0.043	0.811	0.081	0.888	0.922
0	LightGBM_3	0.521	0.054	0.043	0.873	0.083	0.889	0.922

Modélisation

3) Résultats et choix du modèle

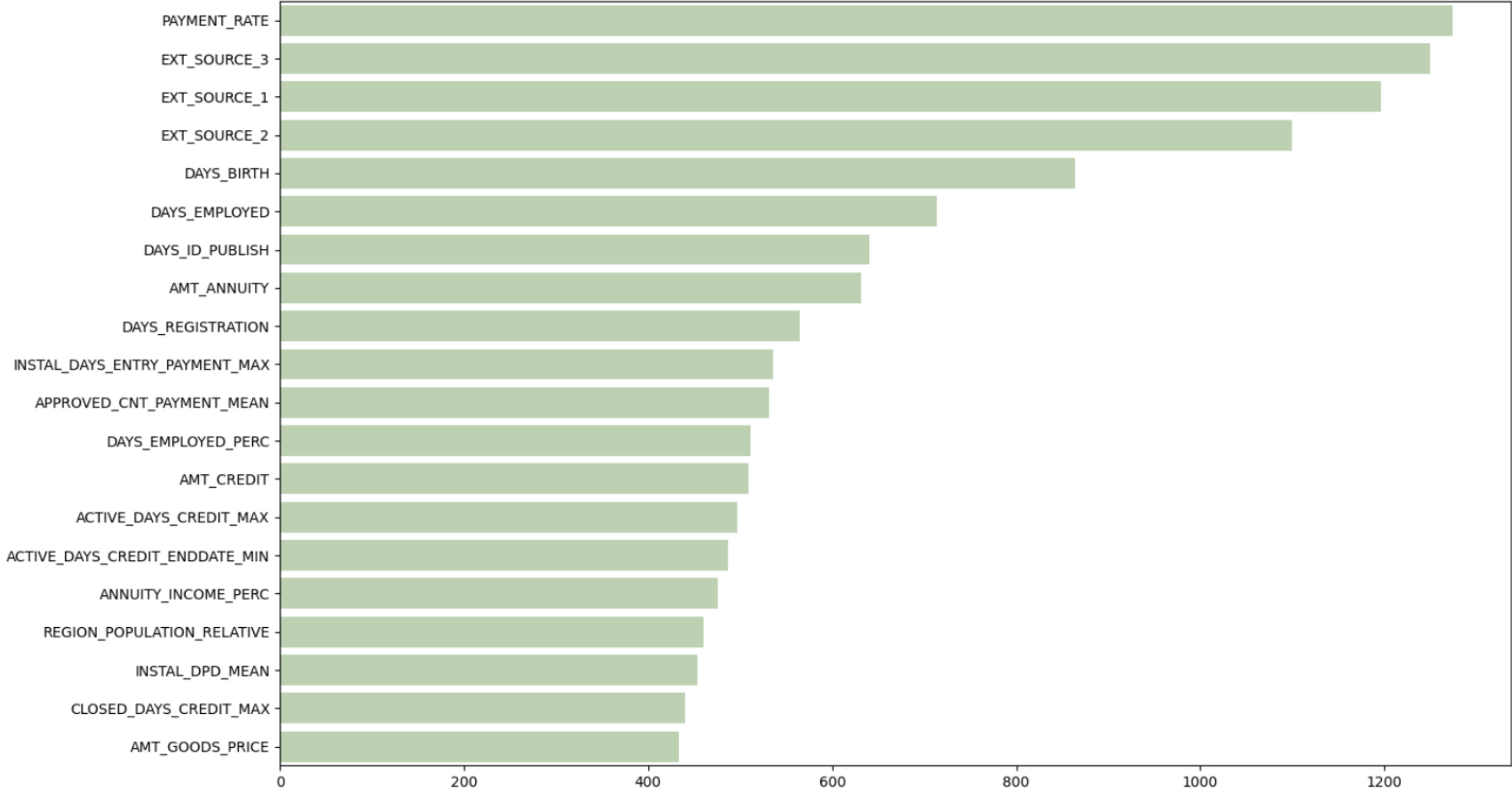
Prediction for : TARGET

	__Model__	__ROC-AUC__	__F-score Beta = 2__	__Recall__	Precision	F1-score	F1-score weighted	Accuracy
0	LightGBM_7	0.531	0.076	0.062	0.884	0.116	0.892	0.924
0	LightGBM_8	0.523	0.057	0.046	0.842	0.087	0.889	0.922
0	LightGBM_4	0.523	0.057	0.047	0.834	0.088	0.889	0.922
0	LightGBM_1	0.521	0.053	0.043	0.811	0.081	0.888	0.922
0	LightGBM_3	0.521	0.054	0.043	0.873	0.083	0.889	0.922
0	LightGBM_5	0.520	0.049	0.040	0.799	0.076	0.888	0.922
0	LightGBM_9	0.520	0.050	0.040	0.845	0.077	0.888	0.922
0	LightGBM_6	0.520	0.049	0.040	0.818	0.076	0.888	0.922
0	LightGBM_0	0.519	0.048	0.039	0.881	0.075	0.888	0.922
0	LightGBM_2	0.518	0.045	0.036	0.860	0.070	0.887	0.922
0	HistBoost_2	0.518	0.049	0.040	0.554	0.074	0.887	0.920
0	HistBoost_1	0.517	0.044	0.036	0.540	0.067	0.886	0.920
0	HistBoost_4	0.517	0.045	0.037	0.543	0.068	0.886	0.920
0	HistBoost_SMOTE_2	0.516	0.042	0.034	0.538	0.065	0.886	0.920
0	HistBoost_3	0.516	0.043	0.035	0.533	0.065	0.886	0.920
0	HistBoost_SMOTE_4	0.515	0.040	0.033	0.527	0.061	0.886	0.920
0	HistBoost_SMOTE_1	0.515	0.041	0.033	0.534	0.062	0.886	0.920

Modélisation

3) Résultats et choix du modèle

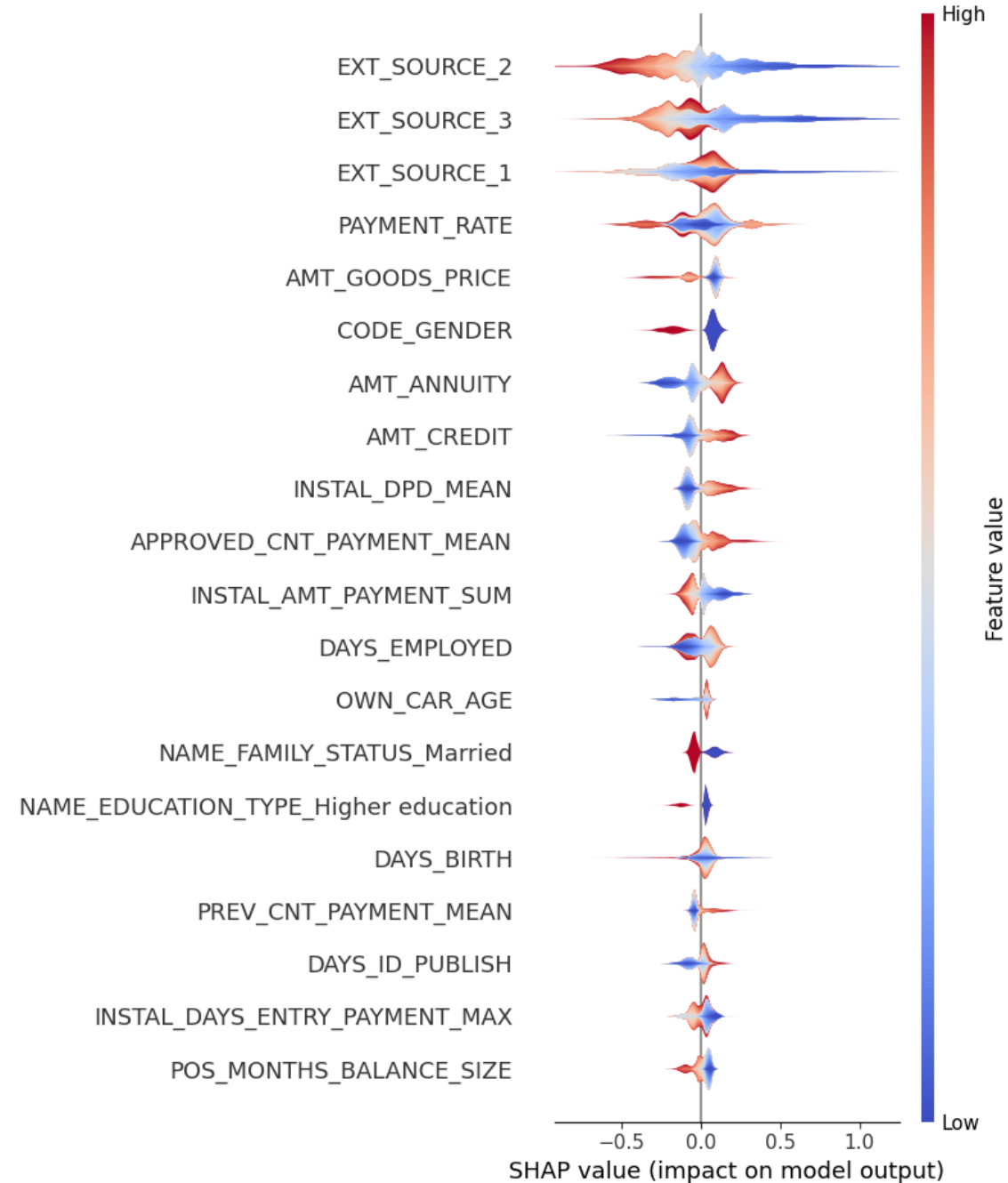
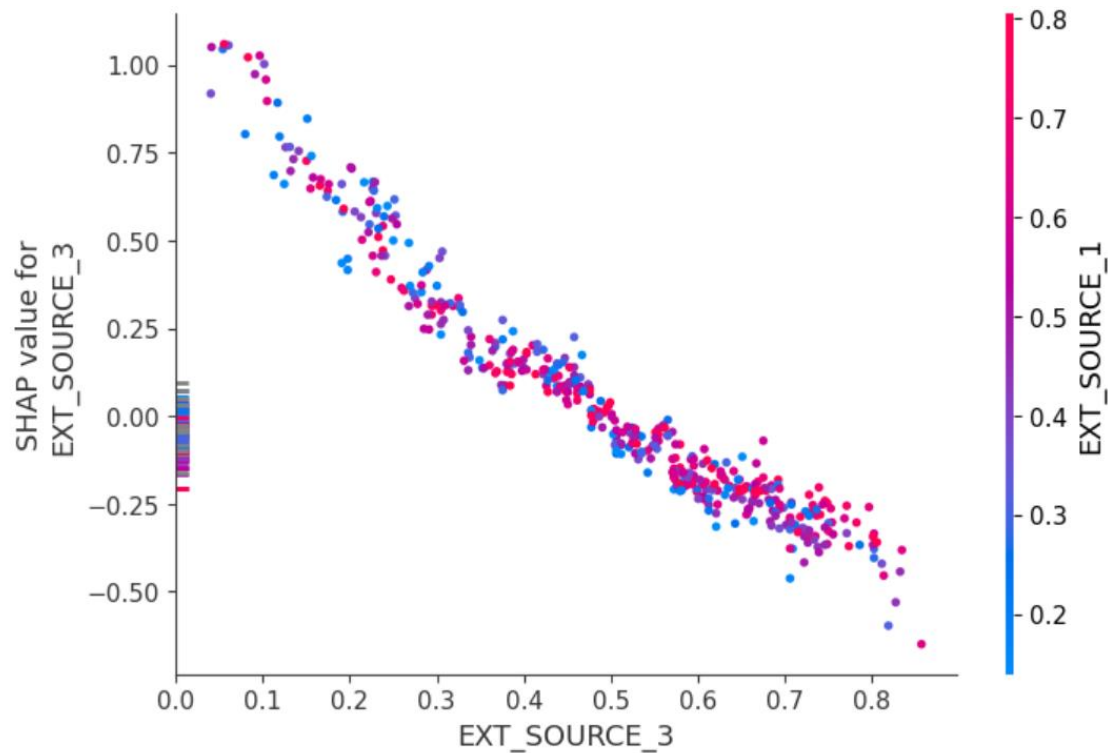
Feature Importance Globale



Modélisation

3) Résultats et choix du modèle

Feature Importance Locale : SHAP



II) Dashboard

Back-end : FastAPI

Front-end : Streamlit

Dashboard

1) Back-end : FastAPI

POST /client_data/ : renvoie les données du client dont on a sélectionné l'ID

POST /predict/ : renvoie la probabilité que le client ne rembourse pas le prêt

POST /shap/ : renvoie les SHAP values du client pour la Local Importance des features

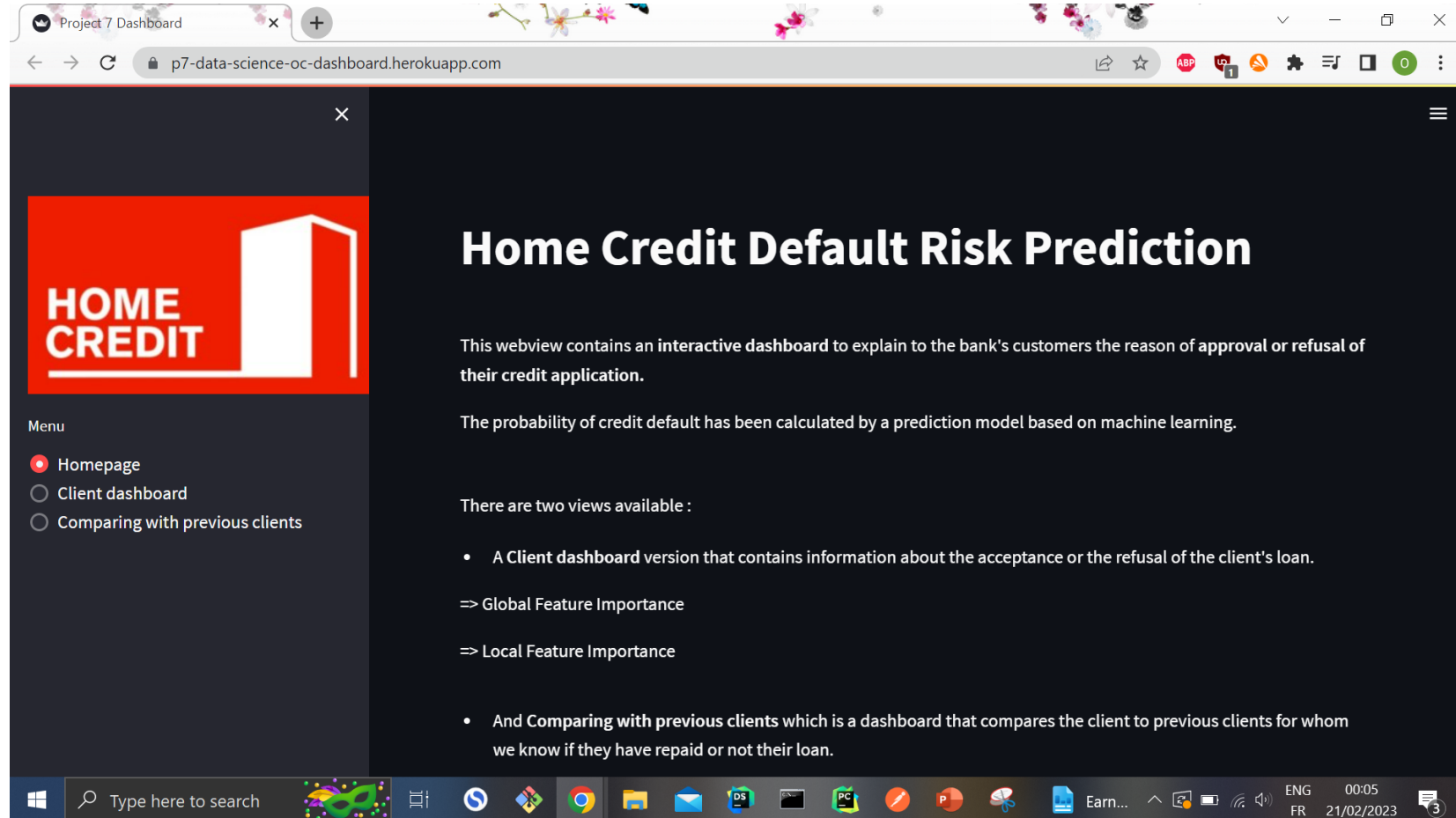
POST /shap_expected/ : renvoie les SHAP values ainsi que l'espérance

POST /feature_importance/ : renvoie l'importance de chaque feature d'après le modèle choisi
(Feature Importance Globale)

Dashboard

2) Front-end : Streamlit

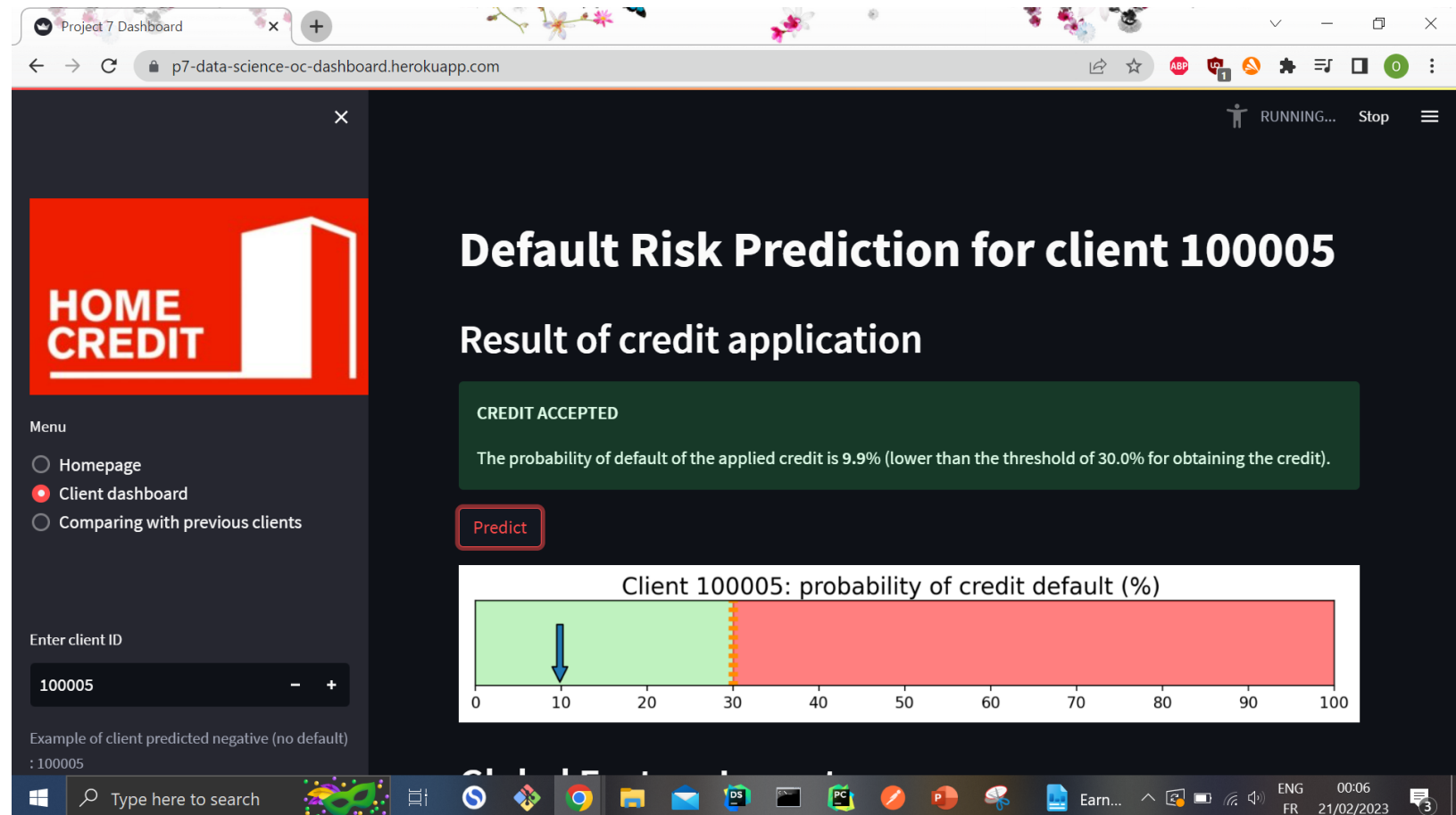
Page d'accueil



Dashboard

2) Front-end : Streamlit

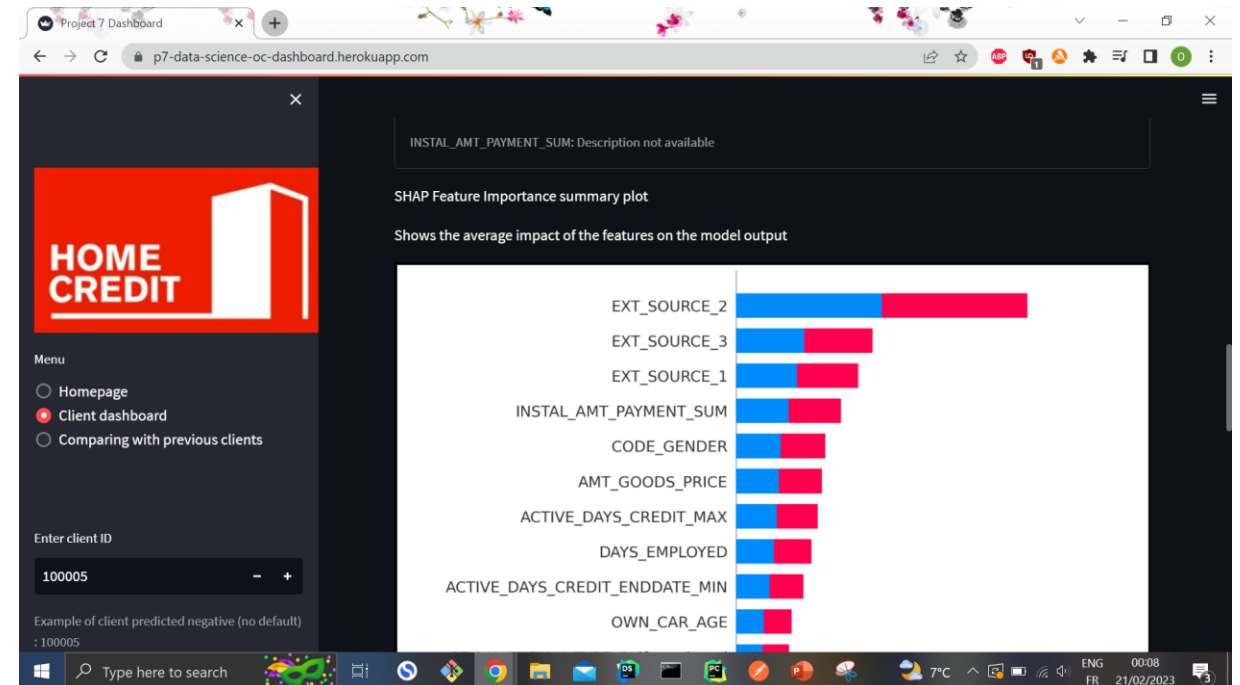
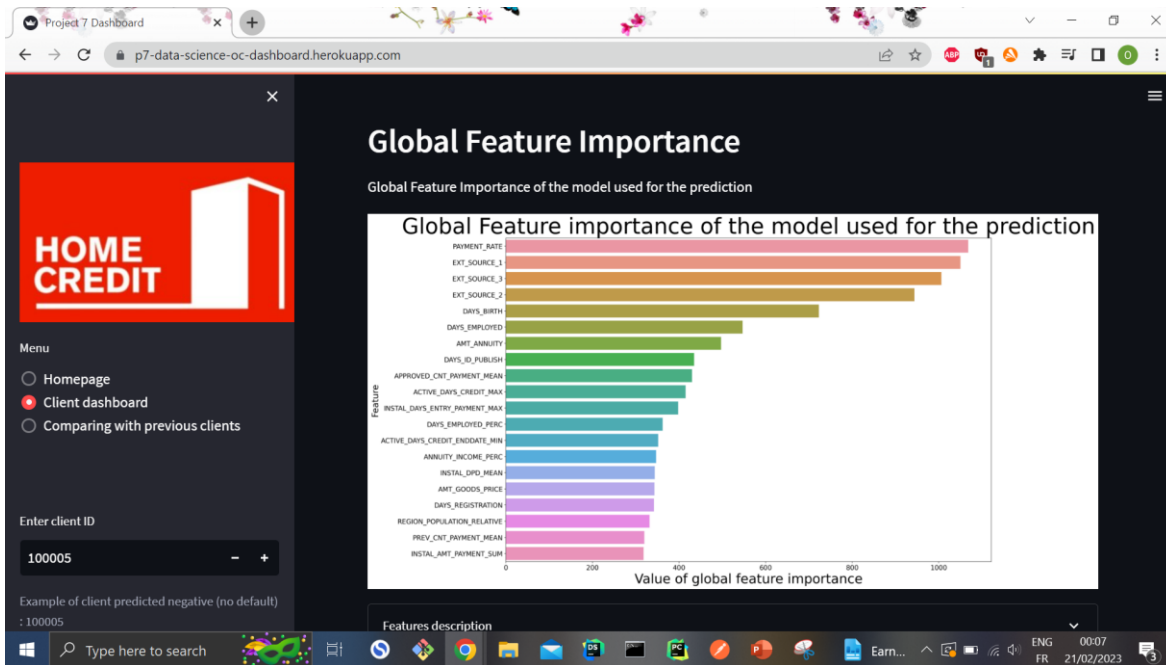
Page où le banquier détermine si le prêt est accordé ou non



Dashboard

2) Front-end : Streamlit

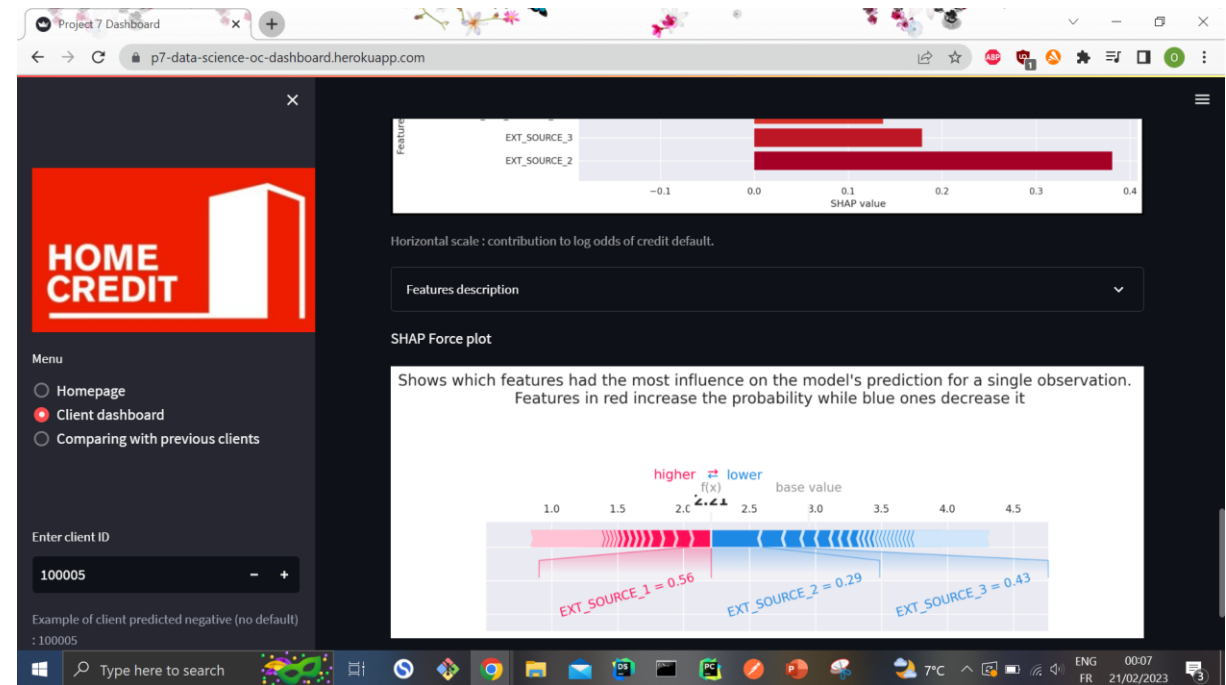
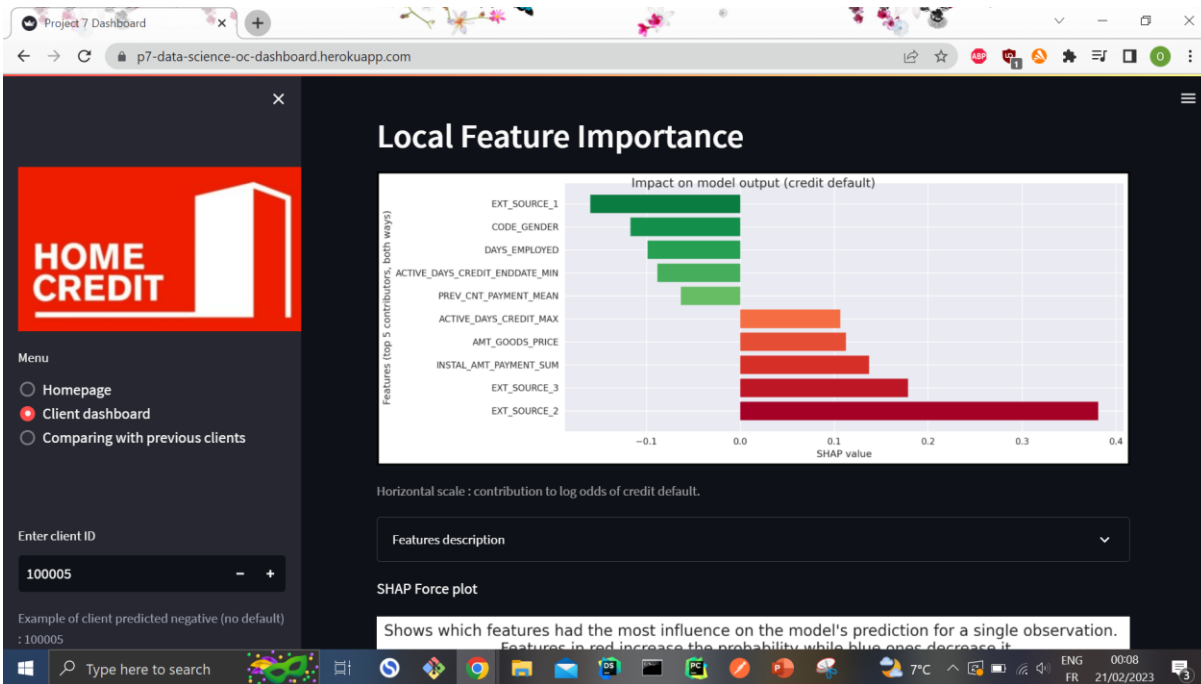
Page où le banquier détermine si le prêt est accordé ou non



Dashboard

2) Front-end : Streamlit

Page où le banquier détermine si le prêt est accordé ou non

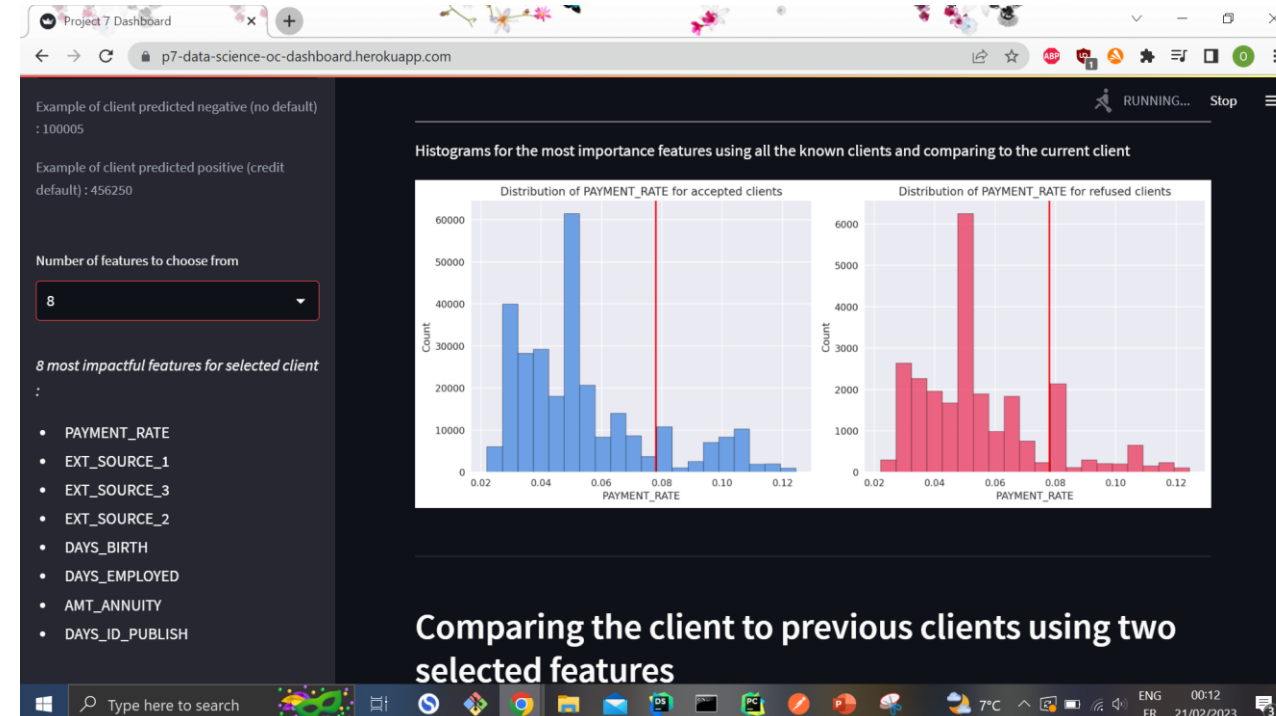
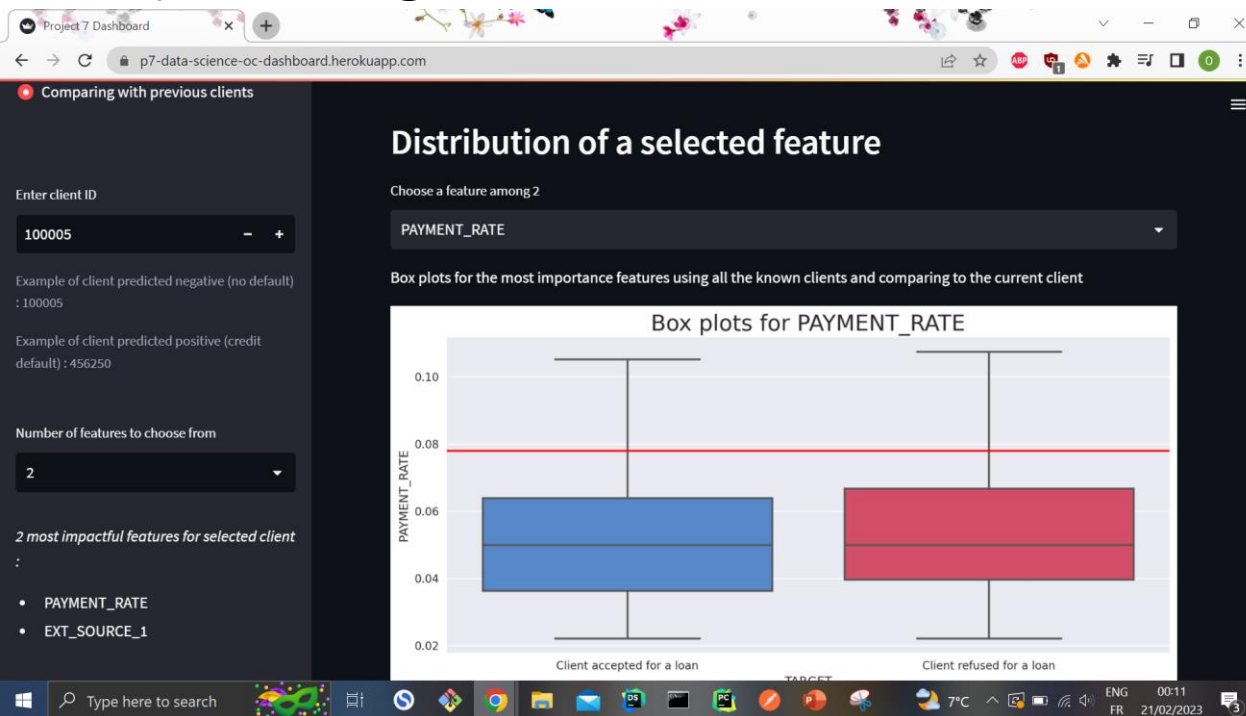


Dashboard

2) Front-end : Streamlit

Page qui permet de positionner le client par rapport aux clients précédents pour lesquels on sait si le prêt a été remboursé ou non

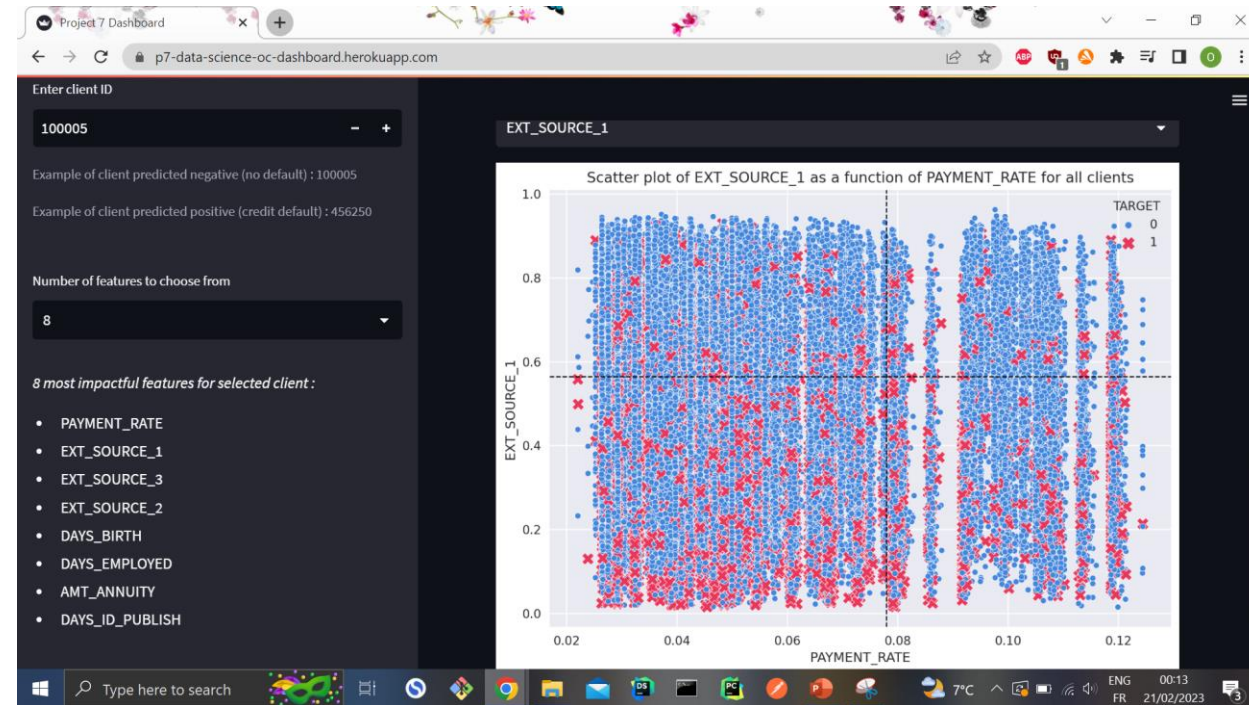
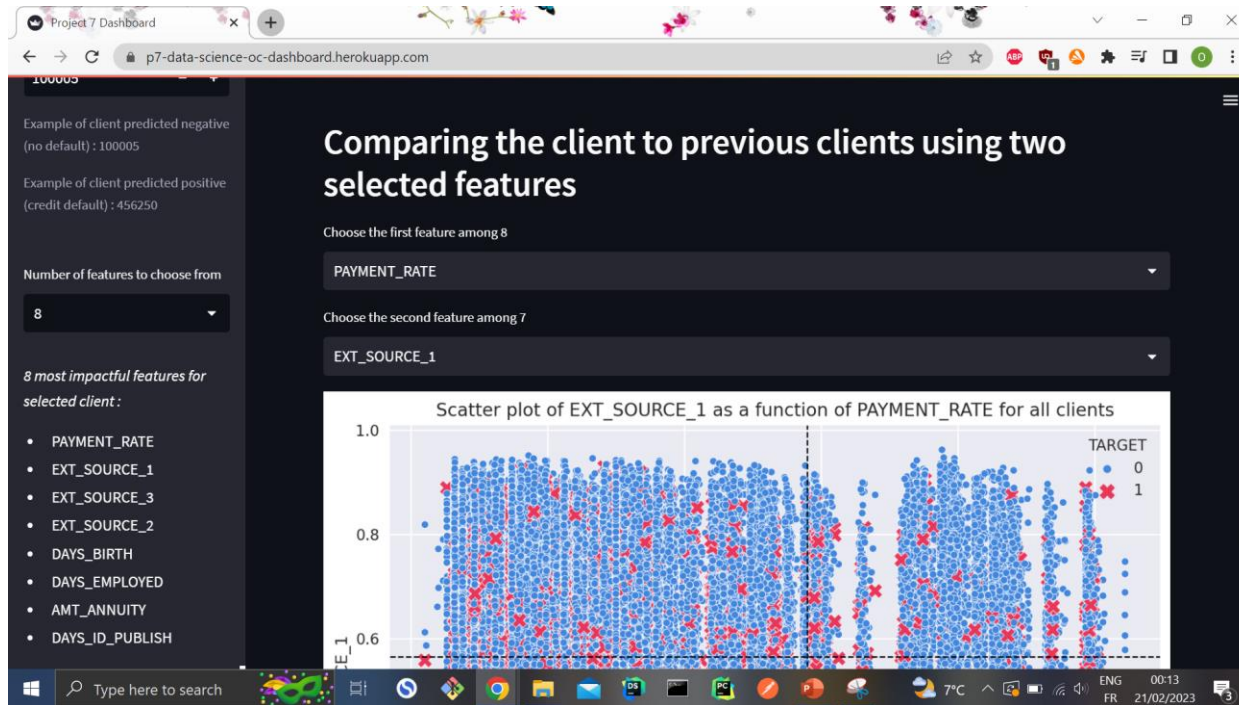
Box plot et Histogrammes



Dashboard

2) Front-end : Streamlit

Page qui permet de positionner le client par rapport aux clients précédents pour lesquels on sait si le prêt a été remboursé ou non



III) Déploiement

Heroku app

Démo

API : <https://p7-data-science-oc-api.herokuapp.com/>

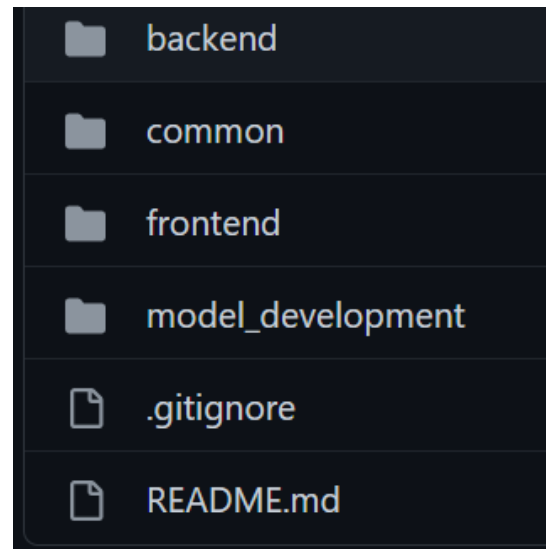
Dashboard : <https://p7-data-science-oc-dashboard.herokuapp.com/>

Déploiement

1) Heroku app

Structure du projet :

- Données (csv et modèle) pour le Back-end et pour le Front-end dans un Bucket de **Google Cloud Storage**
- Téléchargement des fichiers au lancement de l'application web (limite de stockage à 100Mb sur Github)
- Un même repository **Github** pour le **versioning**
- **Deux applications Heroku**



Déploiement

2) Démo

Dashboard : <https://p7-data-science-oc-dashboard.herokuapp.com/>

Conclusion

- Modélisation
- Résultats supérieurs à la baseline (Dummy) mais ils restent assez faibles pour une utilisation directe par la banque.
- Dashboard
- Comprend la prédiction basée sur la probabilité que le client ne rembourse pas le prêt avec une jauge (seuil à 40% basé sur une logique métier)
- Analyse de la feature importance globale et locale
- Graphiques qui permettent de positionner le client par rapports aux autres clients
- Axes d'améliorations :
- Performances du site : temps de chargement long
- Entrainement des modèles assez longs (3h pour le LightGBM avec 10 folds)
- Optimisation des hyper-paramètres : long en temps de calcul (plus d'1h)
- Autre méthode pour rééquilibrer les classe : undersampling par exemple

Merci !

